A case where a spindly two-layer linear network ~~whips~~ decisively outperforms any neural network with a fully connected input layer

Manfred K. Warmuth, Wojciech Kotłowski, Ehsan Amid

Google and Poznan

ML Foundations Seminar

MSR Research

March 11, 2021

Background: Major differences between two families

GD: stochastic gradient descent, backprop, kernel methods, Newton
EG: Winnow, expert algorithms, Boosting, Bayes

Performance of GD linear in d for sparse targets

Performance of EGU linear in $\log d$ for sparse targets

Recent: Square reparameterization trick reintroduced in [GWBNS17]

EGU can be reparameterized as GD:
Reparameterized forms act like EGU (same regret bounds) [AW20a,b]

Thus we can learn sparse targets with GD

Background: Major differences between two families

GD: stochastic gradient descent, backprop, kernel methods, Newton
EG: Winnow, expert algorithms, Boosting, Bayes

Performance of GD linear in d for sparse targets

Performance of EGU linear in $\log d$ for sparse targets

Recent: Square reparameterization trick reintroduced in [GWBNS17]

EGU can be reparameterized as GD:
Reparameterized forms act like EGU (same regret bounds) [AW20a,b]

Thus we can learn sparse targets with GD

My view of Machine Learning was broken

My previous view:

GD - squared Euclidean regularization
==== BIG CHASM ====
EGU - entropic regularization

New view:

GD on neural net with complete input layers
==== ~~BIG CHASM~~ ====
GD on spindly networks

But is it beautiful?
How Beauty Leads Physics Astray by Sabine Hossenfelder

My view of Machine Learning was broken

My previous view:

GD - squared Euclidean regularization
==== BIG CHASM ====
EGU - entropic regularization

New view:

GD on neural net with complete input layers
==== ~~BIG CHASM~~ ====
GD on spindly networks

But is it beautiful?
How Beauty Leads Physics Astray by Sabine Hossenfelder

My view of Machine Learning was broken

My previous view:

GD - squared Euclidean regularization
==== BIG CHASM ====
EGU - entropic regularization

New view:

GD on neural net with complete input layers
==== ~~BIG CHASM~~ ====
GD on spindly networks

But is it beautiful?
How Beauty Leads Physics Astray by Sabine Hossenfelder

Paradigmic sparse linear problem

$$\begin{pmatrix} -1 & -1 & 1 & -1 & -1 \\ 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 \\ 1 & 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \\ -1 \\ 1 \\ 1 \end{pmatrix}$$

$\pm$ matrix random or Hadamard

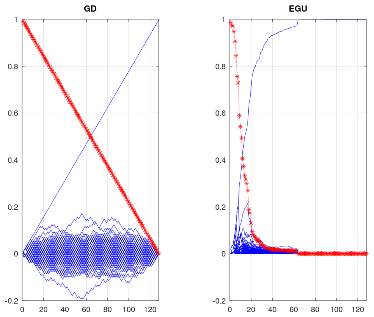
After receiving example $(\boldsymbol{x}_t, y_t)$
and incurring loss $(\boldsymbol{x}_t^\top \boldsymbol{w}_t - y_t)^2$, update:

additive, GD: $w_{t+1,i} = w_{t,i} - \eta \underbrace{2\boldsymbol{x}_{t,i}(\boldsymbol{x}_t^\top \boldsymbol{w}_t - y_t)}_{\text{gradient}}$

multiplicative, EGU: $w_{t+1,i} = w_{t,i} \exp(-\eta 2\boldsymbol{x}_{t,i}(\boldsymbol{x}_t^\top \boldsymbol{w}_t - y_t))$

Linear regression

Major differences in following paradigmatic setup:
128x128 Hadamard matrix
Permuted rows are instances, labels are any fixed column

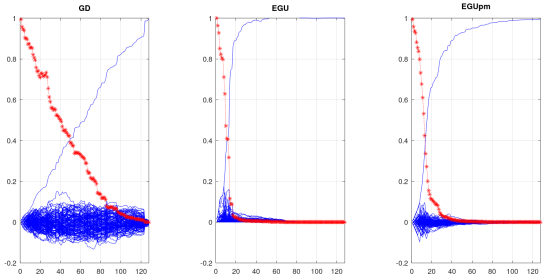


x-axis: $k = 1..128$
y-axis: all 128 weights Average loss when trained on examples $1..k$
$\geq 1 - {}^{k}/_{d}$

Upshot: After half examples, GD has average loss $= {}^{1}/_{2}$
EG family converges in $\log d$ many examples

Linear regression

Major differences in following paradigmatic setup:
128x128 random $\pm$ 1 matrix
Rows are instances, labels are the first column

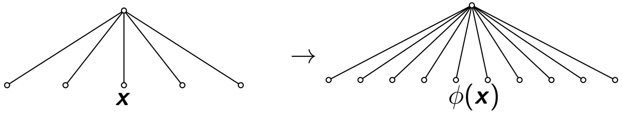


x-axis: $k = 1..128$
y-axis: all 128 weights Average loss when trained on examples $1..k$

Lower bound (experimental) becomes $(1 - {}^{k}/_{d})^2 = 1 - 2\,{}^{k}/_{d} + ({}^{k}/_{d})^2$
Upshot: After half examples, GD has average loss $\approx {}^{1}/_{4}$
EG family converges in $\log d$ many examples

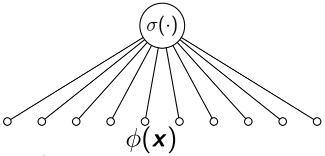
Hardness of Hadamard for GD w. SVD based techniques

- Linear decay of loss remains for GD even if
  - linear neuron with kernel inputs [WV05]



    For k # of examples, $1 - {}^{k}/_{d}$ lower bound remains (SVD based techniques)

  - neuron with any transfer function σ and kernel inputs [DW14]



    Slightly weaker linear lower bound

- NEW: 2 layer complete **linear** neural net, any $\phi(\cdot)$ map, any initialization

$$\geq 1 - {}^{(2k+1)}/_{d}$$

SVD techniques become weak for more than 2 linear layers

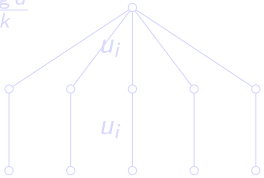
Recent surprise and goal of this talk

Hadamard Problem: Instances are rows of the Hadamard matrix, labels are one of the features (e.g. $\boldsymbol{w} = \boldsymbol{e}_1$)

$$\begin{pmatrix} + & + & + & + \\ + & - & + & - \\ + & + & - & - \\ + & - & - & + \end{pmatrix}$$

Conjecture [DW14]: For the Hadamard problem all neural nets trained w. GD incur loss at least $1 - {}^k/_d$ after seeing k examples

Surprise: Spindly GD trained linear net cracks shifted Hadamard with loss $\leq \frac{\log d}{k}$



Simulates EGU [AW20]

Square reparameterization trick reintroduced in [GWBNS17]

Here: Lower bound of $\geq 1 - {}^k/_d$ for any GD trained net with a **fully connected input layer**

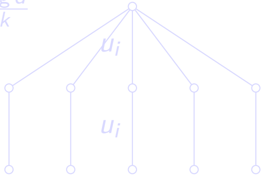
Recent surprise and goal of this talk

Hadamard Problem: Instances are rows of the Hadamard matrix, labels are one of the features (e.g. $\boldsymbol{w} = \boldsymbol{e}_1$)

$$\begin{pmatrix} + & + & + & + \\ + & - & + & - \\ + & + & - & - \\ + & - & - & + \end{pmatrix}$$

Conjecture [DW14]: For the Hadamard problem all neural nets trained w. GD incur loss at least $1 - {}^{k}/_{d}$ after seeing k examples

Surprise: Spindly GD trained linear net cracks shifted Hadamard with loss $\le \frac{\log d}{k}$



Simulates EGU [AW20]

Square reparameterization trick reintroduced in [GWBNS17]

Here: Lower bound of $\ge 1 - {}^{k}/_{d}$ for any GD trained net with a **fully connected input layer**

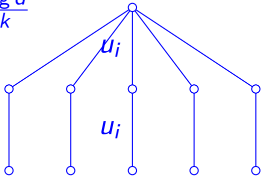
Recent surprise and goal of this talk

Hadamard Problem: Instances are rows of the Hadamard matrix, labels are one of the features (e.g. $\boldsymbol{w} = \boldsymbol{e}_1$)

$$\begin{pmatrix} + & + & + & + \\ + & - & + & - \\ + & + & - & - \\ + & - & - & + \end{pmatrix}$$

Conjecture [DW14]: For the Hadamard problem all neural nets trained w. GD incur loss at least $1 - {}^{k}/_{d}$ after seeing k examples

Surprise: Spindly GD trained linear net cracks shifted Hadamard with loss $\leq \frac{\log d}{k}$



Simulates EGU [AW20]

Square reparameterization trick reintroduced in [GWBNS17]

Here: Lower bound of $\geq 1 - {}^{k}/_{d}$ for any GD trained net with a **fully connected input layer**

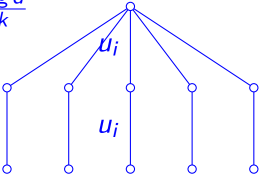
Recent surprise and goal of this talk

Hadamard Problem: Instances are rows of the Hadamard matrix, labels are one of the features (e.g. $\boldsymbol{w} = \boldsymbol{e}_1$)

$$\begin{pmatrix} + & + & + & + \\ + & - & + & - \\ + & + & - & - \\ + & - & - & + \end{pmatrix}$$

Conjecture [DW14]: For the Hadamard problem all neural nets trained w. GD incur loss at least $1 - {}^k/_d$ after seeing k examples

Surprise: Spindly GD trained linear net cracks shifted Hadamard with loss $\leq \frac{\log d}{k}$



Simulates EGU [AW20]

Square reparameterization trick reintroduced in [GWBNS17]

Here: Lower bound of $\geq 1 - {}^k/_d$ for any GD trained net with a **fully connected input layer**

Old versus new lower bounds

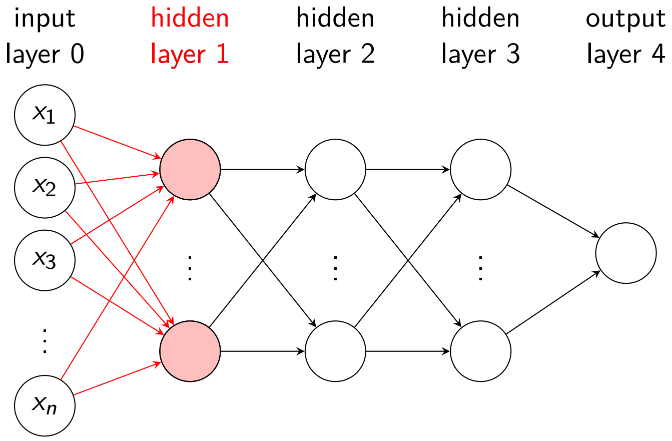
Previous SVD based lower bound technique of [WV05]

- $+$ Holds for any expansion
- $-$ Only for single linear neuron
- $-$ Can't go beyond 2 layer linear
- $-$ Restricted to square loss

NEW: Hold for any GD trained any neural net **of any depth** with a fully connected input layer

- $-$ Does not hold for any expansion (essentially Hadamard or Gaussian)
- $+$ Holds for single target such as constant feature $\boldsymbol{e}_1$
- $+$ Very general losses

Crux: GD trained w. fully connected input layer implies rotational invariance of the predictions



- fully connected input layer with rotation invariant initialization at 1st hidden layer
- weights of first hidden layer trained with GD, any learning rates
- one output node, any initialization
- otherwise any architecture and initialization
- any differentiable transfer functions at the internal nodes
- very general loss function

Minimal loss function assumptions

$L(y, \hat{y})$ differentiable in $\hat{y}$ and
$\min_{\hat{y}} \frac{L(-1,\hat{y})+L(+1,\hat{y})}{2}$ is some positive constant c

Simplifying assumption for the talk: loss is **convex**

Then for proving lower bounds, any randomized algorithm $\hat{y}$ can be turned into a **deterministic algorithm** $\hat{y}_{\text{det}}$:

$$\hat{y}_{\text{det}}(\boldsymbol{x}|(\boldsymbol{X}_{\mathrm{tr}}, \boldsymbol{y}_{\mathrm{tr}})) = \mathbb{E}\left[\hat{y}(\boldsymbol{x}|(\boldsymbol{X}_{\mathrm{tr}}, \boldsymbol{y}_{\mathrm{tr}}))\right]$$

which by Jensen's inequality has loss no greater than the expected loss of $\hat{y}$ on any instance

For simplicity we use **square loss**, i.e. $c = 1$

I) Hardness of learning the constant feature $\boldsymbol{e}_1$

For GD trained neural net with a fully connected input layer

Start with the problem $(\boldsymbol{H}, \mathbf{1})$:
- instances are orthogonal rows of $\underset{d,d}{\boldsymbol{H}}$
- constant target $\boldsymbol{H}\,\boldsymbol{e}_1 = \mathbf{1}$

Too easy

Feed the net randomly sign flipped rows $(s\,\boldsymbol{h}, s)$, for $s = \pm 1$

Net trained with GD. So gradients & weights at input nodes are linear combination of seen instances

Input nodes don't help for predicting label of new orthogonal instances

Also past labels have no info

After seeing k examples, best prediction on $d-k$ new is 0

Therefore, average loss $\geq \frac{d-k}{d} = 1 - \frac{k}{d}$

I) Hardness of learning the constant feature $\boldsymbol{e}_1$

For GD trained neural net with a fully connected input layer

Start with the problem $(\boldsymbol{H}, \mathbf{1})$:

- instances are orthogonal rows of $\underset{d,d}{\boldsymbol{H}}$
- constant target $\boldsymbol{H}\,\boldsymbol{e}_1 = \mathbf{1}$

Too easy

Feed the net randomly sign flipped rows $(s\,\boldsymbol{h}, s)$, for $s = \pm 1$

Net trained with GD. So gradients & weights at input nodes are linear combination of seen instances

Input nodes don't help for predicting label of new orthogonal instances

Also past labels have no info

After seeing k examples, best prediction on $d - k$ new is 0

Therefore, average loss $\geq \frac{d-k}{d} = 1 - \frac{k}{d}$

It's all about rotation invariance of the predictions

A prediction algorithm $\hat{y}(\boldsymbol{x}|(\boldsymbol{X}_{\mathrm{tr}}, \boldsymbol{y}_{\mathrm{tr}}))$ is called **rotation invariant** if

$$\hat{y}(\boldsymbol{U}\boldsymbol{x}|(\boldsymbol{X}_{\mathrm{tr}}\boldsymbol{U}^\top, \boldsymbol{y}_{\mathrm{tr}})) = \hat{y}(\boldsymbol{x}|(\boldsymbol{X}_{\mathrm{tr}}, \boldsymbol{y}_{\mathrm{tr}})), \text{ for any orthogonal matrix } \boldsymbol{U}$$

GD trained neural nets with fully connected input layers are rotation invariant because gradients/weights at input nodes are linear combinations of the training instances

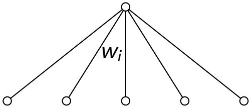
$$\frac{\partial f(\boldsymbol{x} \cdot \boldsymbol{w})}{\partial \boldsymbol{w}} = f'(a)|_{a=\boldsymbol{x}\cdot\boldsymbol{w}}\, \boldsymbol{x}$$

Crux: Any input node that receives an input feature,
must receive all input features

Newton, versions of AdaGrad & Adam, ..., also rotation invariant

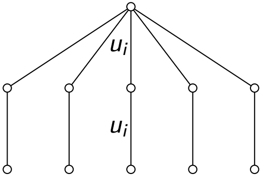
Interlude

EGU on single neuron



$$w_{t+1,i} = w_{t,i} \exp\Big(-\eta\, \boldsymbol{x}_{t,i}(\boldsymbol{x}_t^\top \boldsymbol{w}_t - y_t)\Big)$$

and GD on spindly network



$$u_{t+1,i}^2 = u_{t,i}^2 \Big(1 - {}^{\eta}/_2\, \boldsymbol{x}_{t,i}(\boldsymbol{x}_t^\top \boldsymbol{u}_t \odot \boldsymbol{u}_t - y_t)\Big)^2$$

both decidedly not rotation invariant:

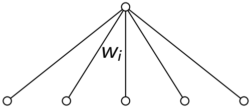
- access individual features
- rank grows exponentially fast

$\dot{\boldsymbol{u}} = -{}^1/_2(\boldsymbol{u}\odot\boldsymbol{u}\cdot\boldsymbol{x} - y)\ \boldsymbol{u}\odot\boldsymbol{x}$ exactly simulates

$$\underbrace{\dot{\log}(\boldsymbol{w})}_{\dot{\boldsymbol{w}}\odot^1/_{\boldsymbol{w}} = 2\dot{\boldsymbol{u}}\odot^1/_{\boldsymbol{u}}} = -\eta\,(\underbrace{\boldsymbol{w}}_{\boldsymbol{u}\odot\boldsymbol{u}}\cdot\boldsymbol{x} - y)\ \boldsymbol{x}$$

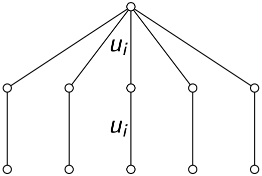
Interlude

EGU on single neuron



$$w_{t+1,i} = w_{t,i}\ \exp\Big(-\eta\ \boldsymbol{x}_{t,i}(\boldsymbol{x}_t^\top \boldsymbol{w}_t - y_t)\Big)$$

and GD on spindly network



$$u^2_{t+1,i} = u^2_{t,i}\ \Big(1 - {}^{\eta}\!/_{2}\ \boldsymbol{x}_{t,i}(\boldsymbol{x}_t^\top \boldsymbol{u}_t \odot \boldsymbol{u}_t - y_t)\Big)^2$$

both decidedly not rotation invariant:

- access individual features
- rank grows exponentially fast

$\dot{\boldsymbol{u}} = -{}^{1}\!/_{2}(\boldsymbol{u}\odot\boldsymbol{u}\cdot\boldsymbol{x} - y)\ \boldsymbol{u}\odot\boldsymbol{x}$ exactly simulates

$$\underbrace{\dot{\log}(\boldsymbol{w})}_{\dot{\boldsymbol{w}}\odot{}^{1}\!/_{\boldsymbol{w}} = 2\dot{\boldsymbol{u}}\odot{}^{1}\!/_{\boldsymbol{u}}} = -\eta\,(\underbrace{\boldsymbol{w}}_{\boldsymbol{u}\odot\boldsymbol{u}}\cdot\boldsymbol{x} - y)\ \boldsymbol{x}$$

Hardness for any rotation invariant algorithm

Initial problem $(\underset{d,d}{\boldsymbol{H}}, \underset{d,1}{\mathbf{1}})$.
Too easy

For each sign pattern $\boldsymbol{s} \in \{+1, -1\}^d$ consider problem $(\mathrm{diag}(\boldsymbol{s})\boldsymbol{H}, \boldsymbol{s})$
Define a rotation matrix $\boldsymbol{U_s}$ as $^1/_{\sqrt{d}}\, \mathrm{diag}(\boldsymbol{s})\boldsymbol{H}$
The predictions of any rotation invariant algorithm on $(\mathrm{diag}(\boldsymbol{s})\boldsymbol{H}, \boldsymbol{s})$ and $(\mathrm{diag}(\boldsymbol{s})\boldsymbol{H}\boldsymbol{U_s}^\top, \boldsymbol{s}) = (\sqrt{d}\boldsymbol{I}, \boldsymbol{s})$ are the same

Now fix $\boldsymbol{s}_{1:k}$. The algorithm receives the same first k training examples for each problem $(\sqrt{d}\boldsymbol{I}, \boldsymbol{s})$. Also since $\boldsymbol{s}_{(k+1):d}$ is chosen uniformly, each of the $d - k$ unseen examples is labeled ± 1 with equal probability. So the best prediction on these $d-k$ examples is 0, incurring square loss at least 1 for each unseen example

Conclusion: Expected average loss on all d examples is at least $^{(d-k)}/_d = 1 - {}^k/_d$

Hardness for any rotation invariant algorithm

Initial problem $(\underset{d,d}{\boldsymbol{H}}, \underset{d,1}{\mathbf{1}})$.
Too easy

For each sign pattern $\boldsymbol{s} \in \{+1, -1\}^d$ consider problem $(\operatorname{diag}(\boldsymbol{s})\boldsymbol{H}, \boldsymbol{s})$
Define a rotation matrix $\boldsymbol{U_s}$ as ${}^1\!/\!\sqrt{d}\,\operatorname{diag}(\boldsymbol{s})\boldsymbol{H}$
The predictions of any rotation invariant algorithm on $(\operatorname{diag}(\boldsymbol{s})\boldsymbol{H}, \boldsymbol{s})$ and $(\operatorname{diag}(\boldsymbol{s})\boldsymbol{H}\boldsymbol{U_s}^\top, \boldsymbol{s}) = (\sqrt{d}\boldsymbol{I}, \boldsymbol{s})$ are the same

Now fix $\boldsymbol{s}_{1:k}$. The algorithm receives the same first k training examples for each problem $(\sqrt{d}\boldsymbol{I}, \boldsymbol{s})$. Also since $\boldsymbol{s}_{(k+1):d}$ is chosen uniformly, each of the $d-k$ unseen examples is labeled ± 1 with equal probability. So the best prediction on these $d-k$ examples is 0, incurring square loss at least 1 for each unseen example

Conclusion: Expected average loss on all d examples is at least ${}^{(d-k)}\!/\!_d = 1 - {}^k\!/\!_d$

Extensions

- Lower bound holds even for rotation invariant initialization of hidden nodes of first layer
  (which must all be connected to all inputs)
- There are versions of the problem where the features are $0/1$ instead of ± 1
- Only minimal requirements on the loss needed:
  $L(y, \hat{y})$ differentiable in $\hat{y}$ and
  $\min_{\hat{y}} \frac{L(-1,\hat{y})+L(+1,\hat{y})}{2}$ is some positive constant c

Not some obscure oscillating function that requires deep networks

$$\begin{pmatrix} -1 & -1 & 1 & -1 & -1 \\ 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 \\ 1 & 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \\ -1 \\ 1 \\ 1 \end{pmatrix}$$

$\pm$ matrix random or Hadamard

- Simple linear functions
- Can't be learned by GD with complete input layer
- But can be learned by spindly
- Essentially for GD, sparse functions seem to require sparse networks

Extensions cont.

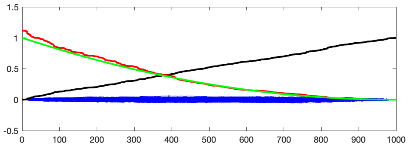
- Only slightly weaker linear lower bound for the case when entries of $\underset{d,d}{\boldsymbol{X}}$ are i.i.d. Gaussian and the target is first column $\boldsymbol{X}\boldsymbol{e}_1$:

  After seeing k examples, the expected average square loss on all d examples is at least $(1 - {}^k/_d)^2 = 1 - 2\,{}^k/_d + ({}^k/_d)^2$
  (so far only for square loss)

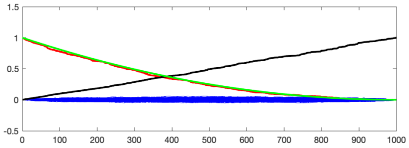
- Experimentally the same lower bound for square loss holds for random ± 1 matrices and single feature targets,
  but no proof yet

In plots

average loss $(1-\frac{k}{d})^2$ good weight remaining 999 weights



$\boldsymbol{X}$ Gaussian, target $\boldsymbol{X}\boldsymbol{e}_1$
LLS provably optimal among rotation invariant algs
expected average loss $(1-\frac{k}{d})^2$



$\boldsymbol{X}$ random ± 1, target $\boldsymbol{X}\boldsymbol{e}_1$
LLS same behavior
no proofs yet

Is there a method here?

Structure of neural net / update
$\longrightarrow$ invariance
$\longrightarrow$ lower bound

	additive	multiplicative
rot. invariance	$\hat{y}(\boldsymbol{Ux} \vert (\boldsymbol{X}_{\text{tr}}\boldsymbol{U}^\top, \boldsymbol{y}_{\text{tr}}))$	$\hat{y}(\boldsymbol{Uxz}^\top\boldsymbol{V}^\top \vert (\boldsymbol{UX}_{\text{tr}}\boldsymbol{Z}_{\text{tr}}^\top\boldsymbol{V}^\top, \boldsymbol{y}_{\text{tr}}))$
	$= \hat{y}(\boldsymbol{x} \vert (\boldsymbol{X}_{\text{tr}}, \boldsymbol{y}_{\text{tr}}))$	$= \hat{y}(\boldsymbol{xz}^\top \vert (\boldsymbol{X}_{\text{tr}}\boldsymbol{Z}_{\text{tr}}^\top, \boldsymbol{y}_{\text{tr}}))$
linear comb.	$\boldsymbol{w} = \boldsymbol{X}_{\text{tr}}^\top \boldsymbol{a}$	$\boldsymbol{W} = \boldsymbol{X}_{\text{tr}}\boldsymbol{C}\boldsymbol{Z}_{\text{tr}}^\top$
hard problem	Hadamard	???
lower bound	$1 - {}^{k}/_{d}$	???

[WKZ14]

The future is open

For Hadamard instances:

- We showed above that GD trained neural nets with a complete input layer cannot learn a **single target** sample efficiently.
- However **any single target feature $\boldsymbol{y}$ can** be learned when inputs are transformed by the map $\phi(\boldsymbol{H}) = \boldsymbol{y}$
- **Conjecture: The d target features cannot be learned with any $\phi(\cdot)$ map when we average over targets**

From XOR to Hadamard - ϕ helps EGU, GD beyond help

$$d\left\{\begin{array}{c} \overbrace{\begin{pmatrix} + & + & + \\ - & + & + \\ + & - & + \\ - & - & + \\ + & + & - \\ - & + & - \\ + & - & - \\ - & - & - \end{pmatrix}}^{\log d} \\ \begin{array}{ccc} b_1 & b_2 & b_3 \end{array} \end{array}\right. \xrightarrow{\phi} \begin{array}{c} \overbrace{\begin{pmatrix} + & + & + & + & + & + & + & + \\ + & - & + & - & + & - & + & - \\ + & + & - & - & + & + & - & - \\ + & - & - & + & + & - & - & + \\ + & + & + & + & - & - & - & - \\ + & - & + & - & - & + & - & + \\ + & + & - & - & - & - & + & + \\ + & - & - & + & - & + & + & - \end{pmatrix}}^{d} \\ \begin{array}{cccccccc} 1 & b_1 & b_2 & b_1 b_2 & b_3 & b_1 b_3 & b_2 b_3 & b_1 b_2 b_3 \end{array} \end{array}$$

ϕ maps a $\log d$ bit pattern $\boldsymbol{b}$ into all $2^{\log d}$ target products

- Products hard to learn from $\log d$ bits by any alg.
- Easy to learn by EGU after expansion with ϕ
- $\phi(\boldsymbol{b}) \cdot \phi(\tilde{\mathbf{b}}) = \sum_{I \subseteq 1..\log d} \prod_{i \in I} b_i \tilde{b}_i = \prod_{i=1}^{\log d} (1 + b_i \tilde{b}_i)$ is $O(\log d)$ [TW02]
- Hard to learn with any kernel (i.e. any feature map ϕ)

See related discussion on learning DNFs w. Winnow [MW98]

Big picture

The Hadamard problem is the exponential expansion of a cryptographically secure problem,
which allows multiplicative updates and their GD reparameterizations to learn this problem sample efficiently

Paradox: GD and LLS optimal of $\phi(\boldsymbol{Y}) = \boldsymbol{Y}$
But that expansion is good for EGU and L_1

Sufficient/insufficient conditions for rotation invariance

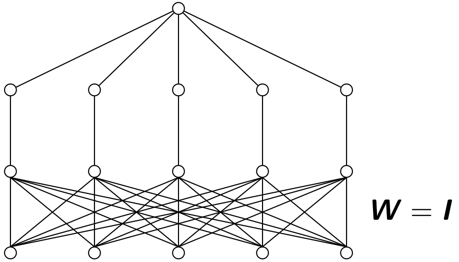
1. Fully connected input layer, rotation invariant initialization @ nodes of 1st hidden layer, all learning rate schedules at these nodes must be invariant under rotating the instances
   Note that dependence of learning rates on dot products $\boldsymbol{w} \cdot \boldsymbol{x}$ and lengths $\|\boldsymbol{x}\|$ is rotation invariant

2. GD, complete input layer, rotation invariant initialization, orthogonal instances, any learning rates @ nodes of 1st hidden
   Since the instances are orthogonal, the coefficients of the past instances (which depend on the learning rates) don't matter when you compute dot product with new instances

3. GD, complete input layer, rotation invariant initialization, non-orthogonal instances, any learning rates @ nodes of 1st hidden layer

- For Gaussian data any linear combination of instances still gets average expected error essentially the same as we already have in the lower bound, so tweaking learning rates for linear predictors does not help

4. Complete input layer, $\boldsymbol{W}^0 = \boldsymbol{I}$, Hadamard instances, $\eta = 0$ for bottom layer, spindly on top
   - Input layer no effect, cracks Hadamard



5. Single linear neuron, Hadamard instances, feature dependent learning rates

Use feature private learning rates to simulate EGU as GD for cracking Hadamard:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + \boldsymbol{N}_t (y_t - \hat{y}_t) \boldsymbol{x}_t \qquad \boldsymbol{N}_t = \operatorname{diag}(\eta_{t,1}, \ldots, \eta_{t,d}),$$

where $\eta_{t,i} = \eta \, \boldsymbol{w}_{t,i}$

Rotation invariance of on-line Newton

Inductive hypothesis: Under transformation $\boldsymbol{x}_t \mapsto \boldsymbol{U}\boldsymbol{x}_t$ for all t we have $\boldsymbol{w}_t \mapsto \boldsymbol{U}\boldsymbol{w}_t$.

Suppose it holds for $\boldsymbol{w}_j$, $j \le t$ and we show it holds for $\boldsymbol{w}_{t+1}$.

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{A}_t^{-1} \nabla_t, \quad \boldsymbol{A}_t = \epsilon \boldsymbol{I} + \sum_{j \le t} \nabla_j \nabla_j^\top, \quad \nabla_t = (\hat{y}_t - y_t)\boldsymbol{x}_t$$

Since $\hat{y}_j = \boldsymbol{w}_j^\top \boldsymbol{x}_j \mapsto \boldsymbol{w}_j^\top \boldsymbol{U}^\top \boldsymbol{U}\boldsymbol{x}_j = \hat{y}_j$, we have $\nabla_j \mapsto \boldsymbol{U}\nabla_j$ for $j \le t$, so $\boldsymbol{A}_t \mapsto \boldsymbol{U}\boldsymbol{A}_t\boldsymbol{U}^\top$. Therefore:

$$\boldsymbol{w}_{t+1} \mapsto \boldsymbol{U}\boldsymbol{w}_t - \eta \boldsymbol{U}\boldsymbol{A}_t^{-1}\boldsymbol{U}^\top \boldsymbol{U}\nabla_t = \boldsymbol{U}\boldsymbol{w}_{t+1}$$

Versions of Adagrad

- Diagonal version:

$$\boldsymbol{w}_{t+1,i} = \boldsymbol{w}_{t,i} - \frac{\eta}{\sqrt{\epsilon + \sum_{j \le t} \nabla_{j,i}^2}} \nabla_{t,i}$$

  Clearly not rotation-invariant, but when $\boldsymbol{x}_{t,i} = \pm 1$ for all t, i, then $\nabla_{j,i}^2 = (y_t - \hat{y}_t)^2$, i.e. the effective learning rate is shared among all coordinates and rotation invariance holds.

- Full version:

$$\boldsymbol{w}_t = \boldsymbol{w}_t - \eta \sqrt{\boldsymbol{A}_t} \nabla_t, \qquad \boldsymbol{A}_t = \epsilon \boldsymbol{I} + \sum_{j \le t} \nabla_j \nabla_j^\top$$

  Rotation invariant, proof analogous to On-line Newton

Adam

- Adagrad with momentum and exponentially decaying memory on the past
- Essentially the same arguments as for diagonal Adagrad: generally not rotation invariant, but for ± 1 valued data it is rotation invariant.

[VAW] and [FW]

$$\boldsymbol{w}_t = \boldsymbol{A}_t^{-1} \sum_{j \le t-1} y_j \boldsymbol{x}_j, \qquad \boldsymbol{A}_t = \epsilon \boldsymbol{I} + \sum_{j \le t} \boldsymbol{x}_j \boldsymbol{x}_j^\top.$$

VAW is also rotation invariant: under transformation $\boldsymbol{x}_t \mapsto \boldsymbol{U}\boldsymbol{x}_t$ for all t, we clearly have $\boldsymbol{A}_t \mapsto \boldsymbol{U}\boldsymbol{A}_t\boldsymbol{U}^\top$, so $\boldsymbol{A}_t^{-1} \mapsto \boldsymbol{U}\boldsymbol{A}_t^{-1}\boldsymbol{U}^\top$, and thus

$$\boldsymbol{w}_t \mapsto \boldsymbol{U}\boldsymbol{A}_t^{-1}\boldsymbol{U}^\top \sum_{j \le t-1} y_j \boldsymbol{U}\boldsymbol{x}_j = \boldsymbol{U}\boldsymbol{A}_t^{-1} \sum_{j \le t-1} y_t \boldsymbol{x}_j = \boldsymbol{U}\boldsymbol{w}_t$$

But is it beautiful?

How Beauty Leads Physics Astray by Sabine Hossenfelder

We all have become Physicists.
We don't understand how neural networks work

We can only study this natural phenomenon